Introduction to Software Engineering.

Software, IEEE, Layered Technology, Generic view of process, frame work, Activities, Umbrella Activities, capability Maturity model (cmm).

Software:- Software is a collection of program, data and instructions that performs specific tasks (or) functions. Software can be broadly divided into two types: 1. SYSTEM SOFTWARE
2. APPLICATION SOFTWARE

1. SYSTEM SOFTWARE:-

Operating system, device drivers and utility programs. That provides the basic functionality for a computer device.

2. APPLICATION SOFTWARE:-

Programs that perform specific task such as gaming, web browsing and word processing.

SOFTWARE ENGINEERING:-

Software engineering is a process of developing, Testing and deploying. Computer Applications to solve real world problems by adhering to a set of engineering principles and best practises.

Introduction to Software Engineering.

Software, IEEE, Layered Technology, Generic View of process, frame work, Activities, Umbrella Activities, capability Maturity model (cmm).

Software:- Software is a collection of program, data and instructions that performs specific tasks (or) functions. Software can be broadly divided into two types: 1. SYSTEM SOFTWARE

2. APPLICATION SOFTWARE

1. SYSTEM SOFTWARE:-

Operating system, device drivers and utility programs. That provides the basic functionality for a computer device.

2. APPLICATION SOFTWARE:-

Programs that perform specific task such as gaming, web browsing and word processing.

SOFTWARE ENGINEERING:

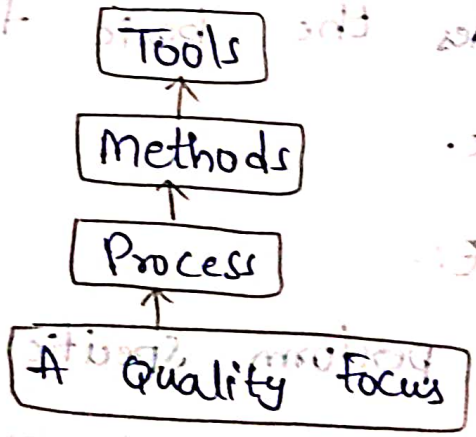
Software engineering is a process of developing, Testing and deploying. Computer Applications to solve real world problems by adhering to a set of engineering principles and best practises.

IEEE :

Indian standard 610.12-1990 defines software engineering as the application of a systematic, disciplined which is a computable approach for the development, Operations and maintenance of software.

LAYERED TECHNOLOGY:

To develop a software we need to go from one layer to another layer. All the layers are connected and each layer demands the fulfillment of the previous layer.



A Quality focus:-

It provides integrity that means providing security to the software. So that data can be accessed by authorized person no outsiders can access the data.

process :- It is the foundation (or) base layer of software engineering. It covers all the activities, actions and tasks required to be carried out for software development.

Methods:-

During the process of software development it will answer all the "how to do" questions. method has information about all the tasks which includes, communication requirement Analysis, design modelling, progoam Construction testing and support.

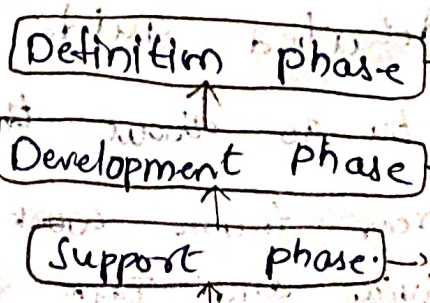
Tools:-

Tools provide a self operating system for process and methods (information created by one tool can be used by another tool).

Generic View of process:-

Generic process model is an abstraction of software development process.

It is used in most of the softwares, because it provides base for them.



Definition phase: focused on software project planning requirements, Analysis. what, how, when, why not all the questions related words are in this phase.

Development phase:- It focuses on how Questions.

* Software Design, Code generation and Software Testing.

SUPPORT PHASE:-

* Support phase is focused on change.

* change due to the enhancement, what by the Customer requirement.

Change is of four types in support phase.

Correction:- Corrective maintenance change the software to correct defect.

Adaption:- It modifies the software.

Enhancement:- Customer/User will recognise Additional functions that will provide benefits.

prevention:- Software reconstruction.

FRAMEWORK ACTIVITIES:-

* Every process needs a particular order (or) sequence that must be followed to keep things Right

* It illustrates the detailed guide within a framework and how to go about the application.

* The Software Engineer frame work include process Activity, umbrella Activity and Task sets.

Software process

Process framework

Umbrella Activities.

Framework Activity #1

Software Engineering action #1.1

Task sets

- Work Task
- Work Products
- Quality Assurance points
- Project milestones.

Software Engineering Action #1.k

Task Sets

- Work Task
- Work products
- Quality Assurance points
- project milestones.

Framework Activity # n

Software engineering Action #n.1

Task sets

- Work Task
- Work Products
- Quality Assurance points
- Project milestones.

Software engineering Action #n.k

Task set

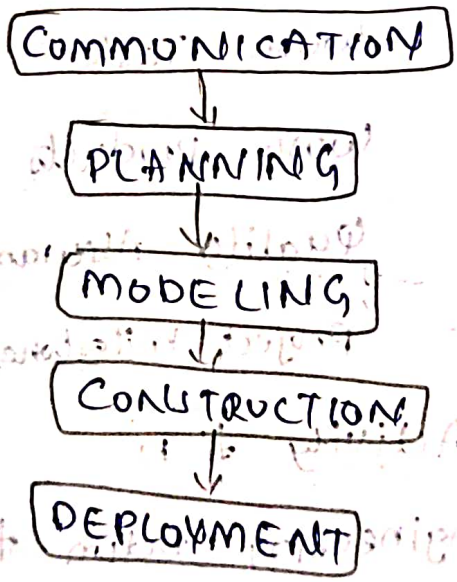
- Work Task
- Work Products
- Quality Assurance points.

It demonstrates

1. Task :- specified goal that has to be elaborated
2. Action :- What must be done to Acheive the elaborated task.
3. Activities :- All the minor (or) related procedures are illustrated to Acheive the Action and the task.

Process framework Activities :-

Processed framework is a base structure where all the Activities are described in detailed to Acheive a task.



1. Communication :- Necessary to know the Actual demand of the Client.
2. planning :- Join a map for reduced the Complications of development.

3. Modeling:- A model is created according to the client for better understanding.

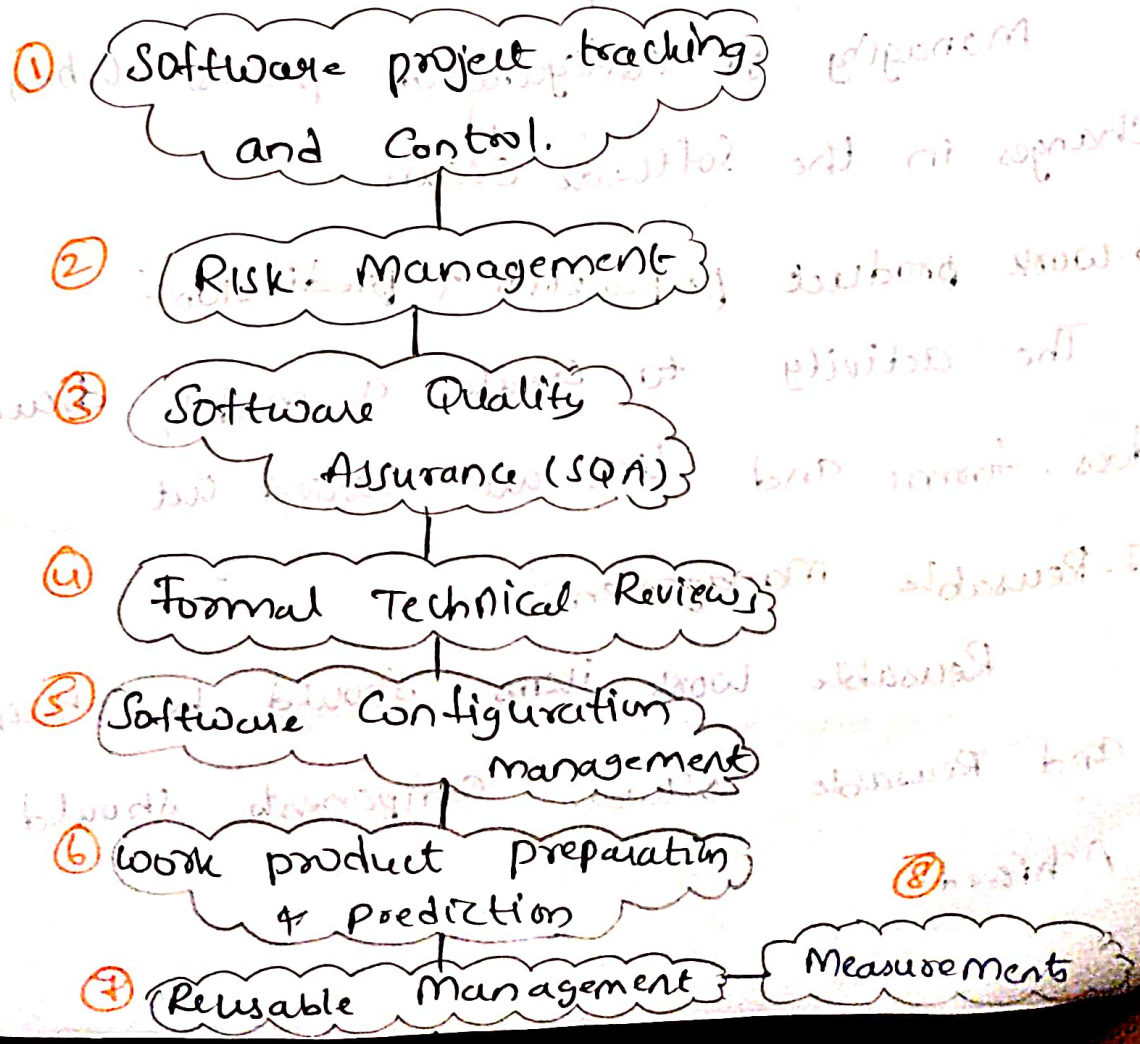
4. Construction:- Coding and testing will be done for the given problem.

5. Deployment:-

Delivery of software to the client for evaluation and feed back.

Umbrella Activity:-

Umbrella Activity is equally important, to take measurements to track the development of applications and make the necessary changes for any improved that might be needed.



1. Software project Tracking and Control:-

Take actions to keep the project on time by Comparing the projects process against plan

2. Risk Management:-

The Risk that might affect the project outcome (or) Quality can be analysed.

3. Software Quality Assurance:- perform Actions to ensure the products Quality.

4. Formal Technical Reviews:-

At each level of process errors are evaluated and fixed.

5. Software Configuration Management:-

Managing of configuration process. When any changes in the Software occurs.

6. Work product preparation & prediction:-

The activity to create a model, document, docs, forms and lists are carried out.

7. Reusable Management:-

Reusable work items should be backed up and Reusable Software Components should be Achieved.

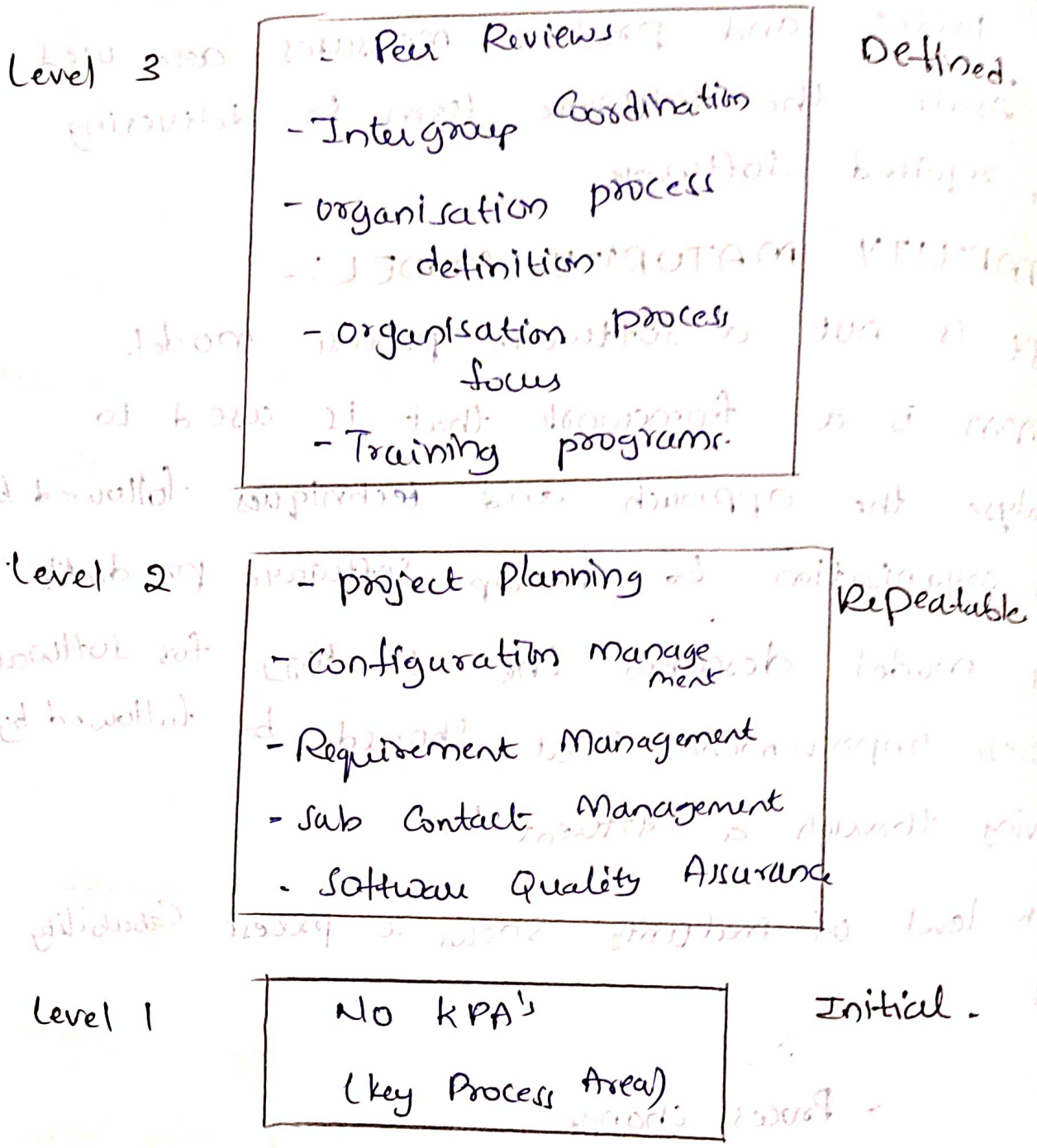
8. Measurement:-

project and product measures are used to Assist the software team in delivering the required software.

CAPABILITY MATURITY MODEL:-

* It is not a software process model.
* CMM is a framework that is used to analyse the approach and techniques followed by the organization to develop software products.
* This model describes the strategy for software process improvement. That should be followed by moving through a different level.
* Each level of maturity shows a process capability level.

Level 5	- Process change management - Technology change management - Defect prevention	optimizing
Level 4	- Software Quality Management - Quantitative management	managed



The five levels of CMM as follow.

LEVEL 1: INITIAL

* No key process Area. is defined.
* A few process are defined and success depe on individual effort.

LEVEL 2: Repeatable.

* This levels helps to keep track of cost, Sched and functionality.

LEVEL 3:- DEFINED

The software process is standardized, documented and integrated into organisation wide software process. This level includes all characteris defined for level 2.

LEVEL 4:- MANAGED

Both the software process and products are Quantitatively understood and controlled using detailed measures.

This level includes all the characteristic defined for level 3.

LEVEL 5: OPTIMIZING

Continuous process improvement is enabled by Quantitative feedback from the proces and from testing innovative ideas and Technologies.

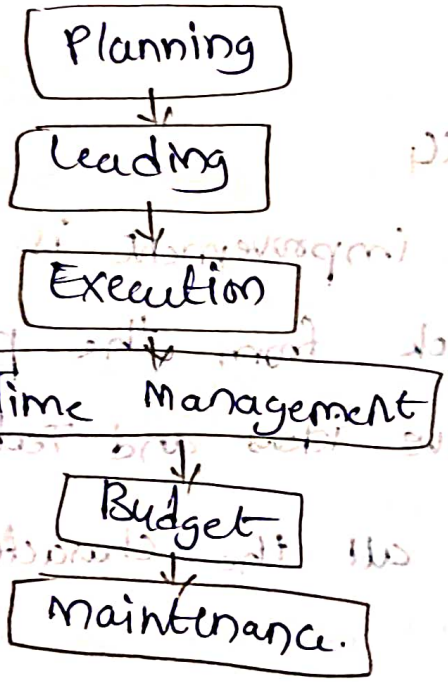
This level includes all the characteristics defined for level 4.

2. SOFTWARE PROCESS MODEL.

SPM, Software development lifecycle, SDLC Activity waterfall model.

SPM: [Software project management]

* Software project management is an art and discipline of planning and supervising.
* project defines as a set of inputs and outputs which are required to achieve the goal.



1. Planning:-

A software project Manager layouts the complete project blueprint.

2. Leading:-

A software project Manager brings together and leads a team of engineers, programmers, designs and data scientists.

3. Execution:-

Each stage of the project is completed successfully by measuring progress and monitoring to check. How the function and status report are all parts of success.

4. Time Management:-

It is about allocating resources to complet task on time while staying within the approved budget.

5. Budget:-

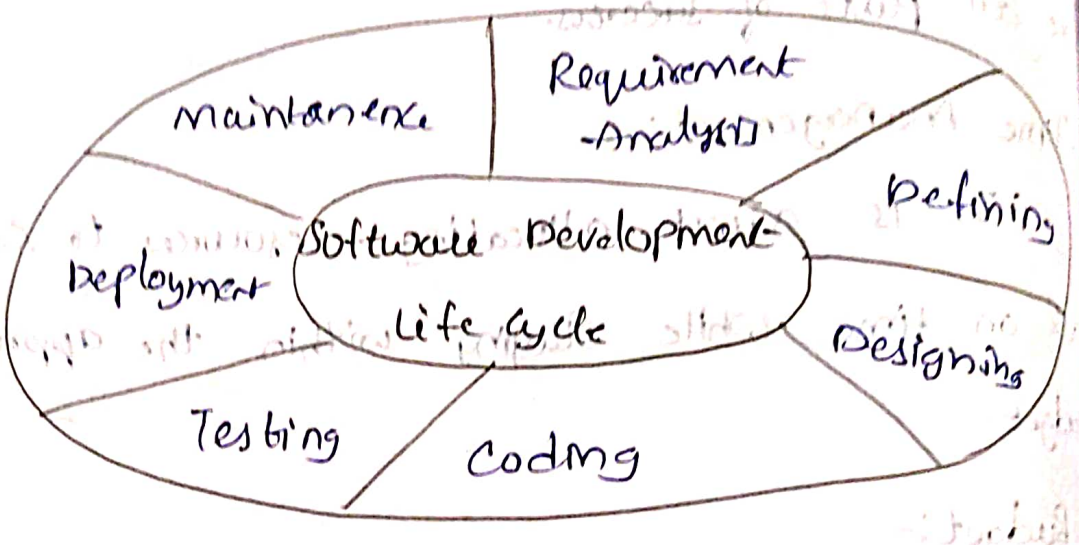
How to efficiently manage the money aspects of projects to avoid running into a financial crunch. later on in the project.

6. Maintenance:-

Software project management emphasizes Continuous product testing to find and replace defect earlier.

Software Development life cycle Model:

SDLC is the structured process that enables the production of high quality, low cost software in the shortest production time.



Stage 1 :- PLANNING AND REQUIREMENT ANALYSIS

* Planning for Quality Assurance Requirement.
* Identification of risk associated with the project also done in this stage.

Stage 2:- DEFINING REQUIREMENTS:

Software Requirement Specification Document which contains all the product Requirements to be constructed and develop during the project lifecycle.

STAGE 3:- DESIGNING THE SOFTWARE

It brings down all the knowledge of requirements, Analysis and design of the software

STAGE 4:- DEVELOPING A PROJECT.

The Actual development is begin and programming is built.

Developers will be using tools like debuggers, compilers. Interpeters are used to develop and implement the Code.

STAGE 5: TESTING

In this stage unit Testing, integration testing, System Testing and acceptance testings are done.

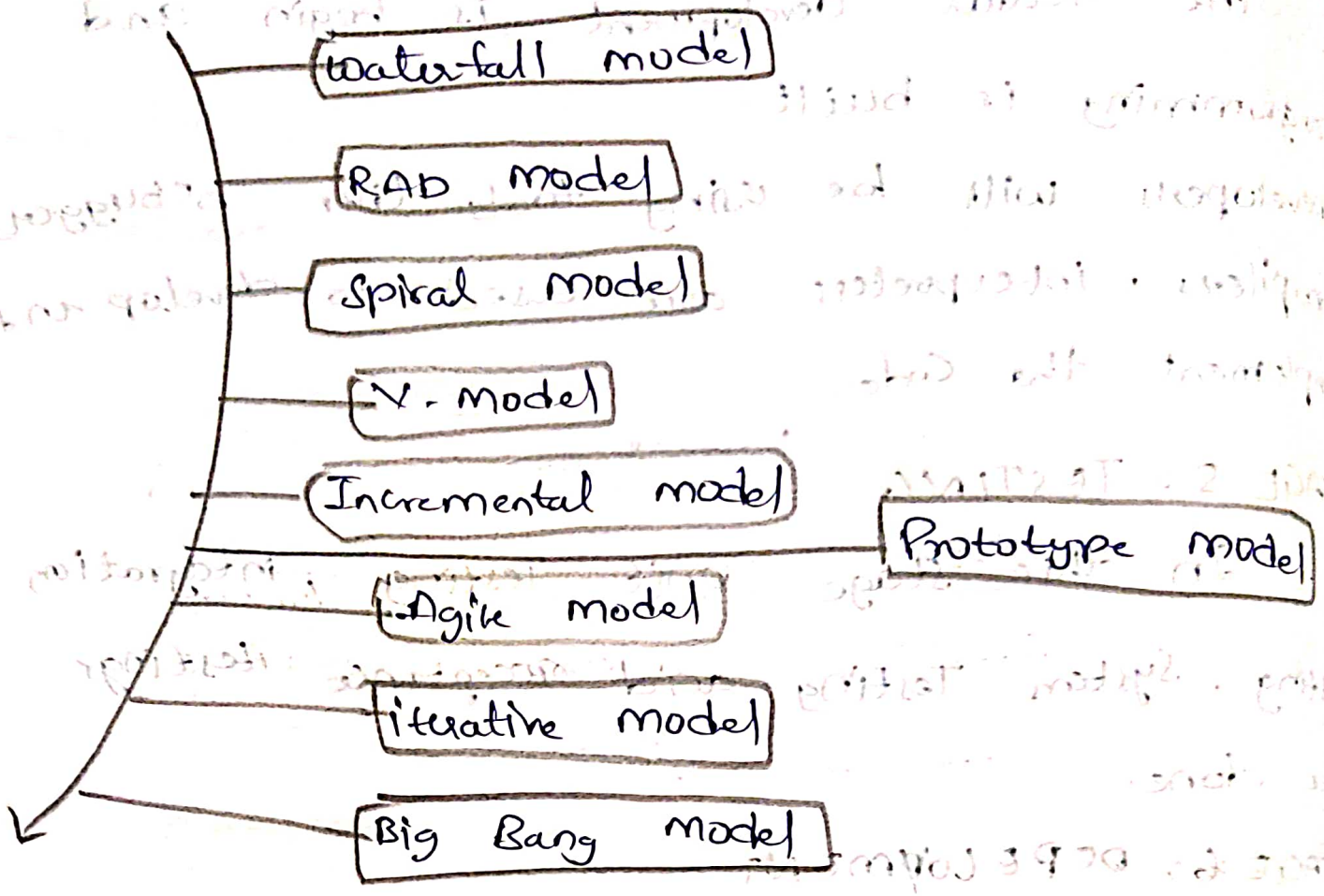
STAGE 6:- DEPLOYMENT

Once the software is certified as no bugs errors are stated then it is deployed.

STAGE 7:- MAINTENANCE.

once when the client starts using the developed system then the real issues comes up and requirements to be solved from time to time.

SDLC MODELS



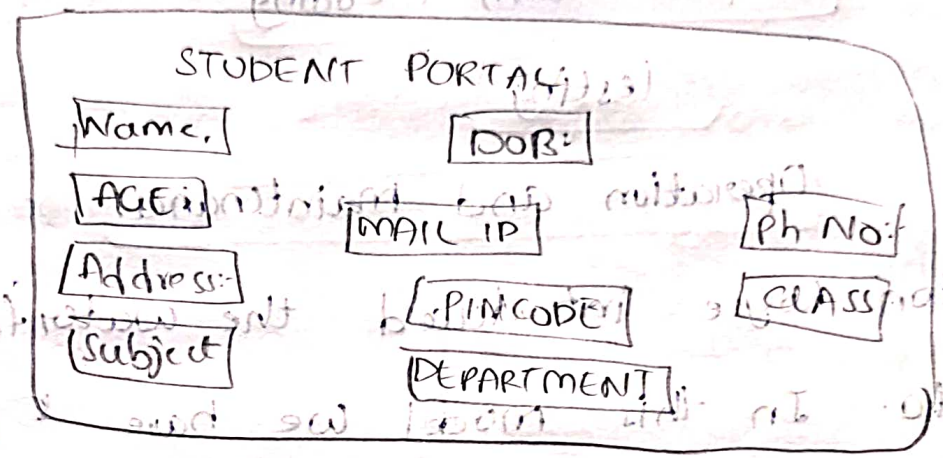
* Consider you got recruited in a Zoho Compan as a software developing manager. First deliverab needs based on software Development life cycle model.

* SDLC model is the structured process that enables the production of high Quality, low cost software in the shortest possible time.

* In this students portal software the requirem and need of the software is to get the information about students Quickly with the key points.

2. The defined requirements needed in the software of student portal are Name, Age, Date of birth, State, Country, Register Number, Mother Name, Father Name, Email.id, subjects, Results, Marks etc.

3. It brings all the knowledge of design portal.



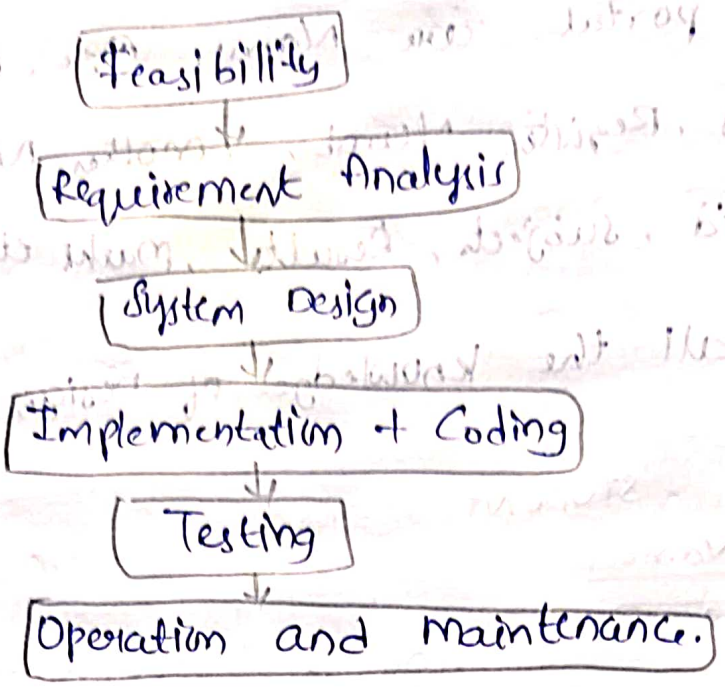
4. Here in this software of students portal we/I used tool like Compiler to compile and execute the code.

5. In this software I used one of the testing method like Unit Testing to check the errors.

6. If there are no errors and bugs in the program coding then the software is certified.

7. Client will be happy receiving successful program code software and if any real time problems comes in the future we need to correct it on time.

1. Waterfall Model.



* Winston Royce introduced the waterfall model in 1970. In this model we have 5 phases.

* Waterfall model is a linear, sequential, approach to the software development life cycle that is popular in software engineering & product development.

* Waterfall model uses logical preparation of SDLC steps for the project.

* Waterfall model are used to develop enterprise application like Customer relationship management system (CRM), Human Resource Management System (HRMS), supply chain management system, Inventory management system, point of scale system for

1. REQUIREMENT ANALYSIS AND SPECIFICATION PHASE

Aim of this phase is to understand exact requirement of the customer and to the document that properly.

* Both the customer and software Developer work together. In this phase large document are called as software Requirement specification.

2. DESIGN PHASE:-

It defines the overall software architecture togetha with high level and detailed design.

+ All this work is documented as a software design document (SDD).

3. IMPEEMENTATION AND UNIT TESTING:-

* Design is implemented.

+ If SDD is complete implementation or Coding phase proceeds smoothly.

UNIT TESTING:-

Unit Testing is the software developing process in which the smallest testable parts of the applications called units, individually and independently done for proper operation.

INTEGRATION AND SYSTEM TESTING:-

This phase is highly crucial as the Quality of the end product is determined Testing carried out. Better Ouput will lead to satisfied Customer low maintenance Cost and acc results.

OPERATION AND MAINTENIANCE PHASE:-

Maintenance is the task performed by eng year once software has been delivered to the Customer Installed and Operational.

WHEN TO USE WATERFALL MODEL?

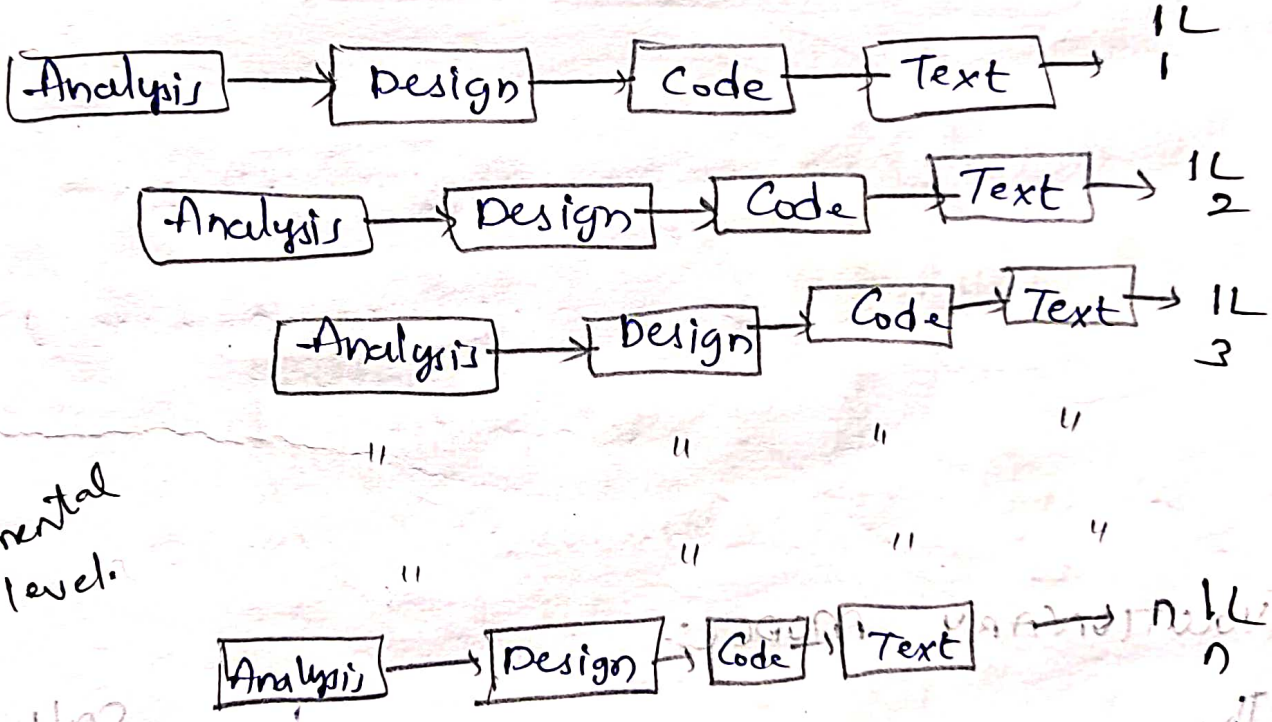
1. When the project is short.
2. When the requirement are Constant and not changing regularly.
3. Way the tools and technology are used is Constant and not changing.
4. The situation is Clam.

Advantages

Disadvantages.

2. INCREMENTAL LIFE CYCLE MODEL:

It was developed by European Space Agency in 1991. Incremental model makes a "unrealistic" assumption that the system. As well as software Requirements remains stable.



1L = incremental level.

1st Incremental level:-

In the first / increment phase basic file management editing and document production.

2nd Incremental level:-

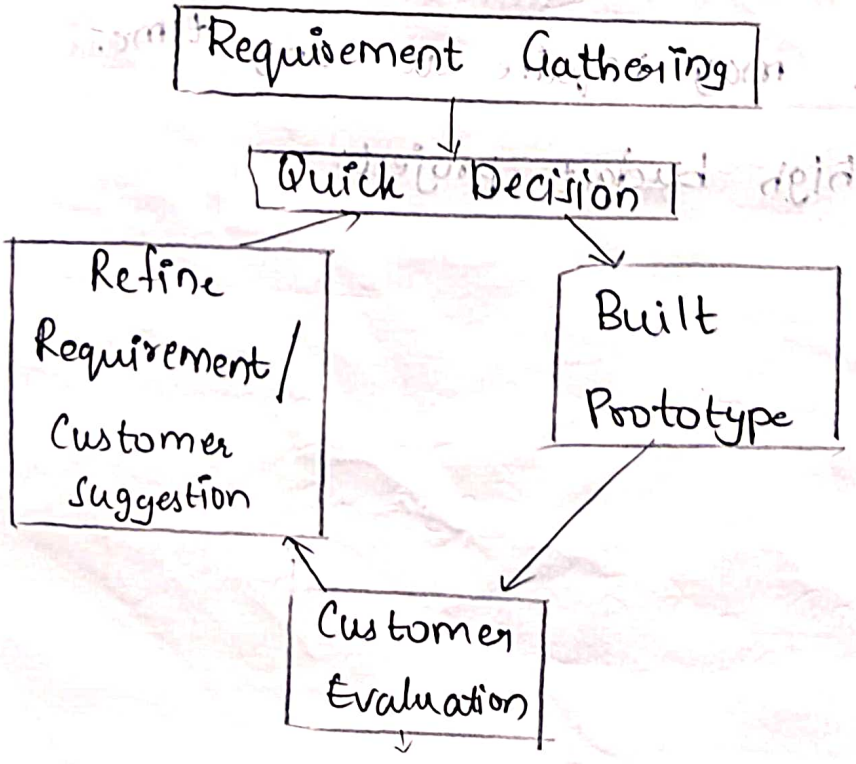
Spelling And grammar checking.

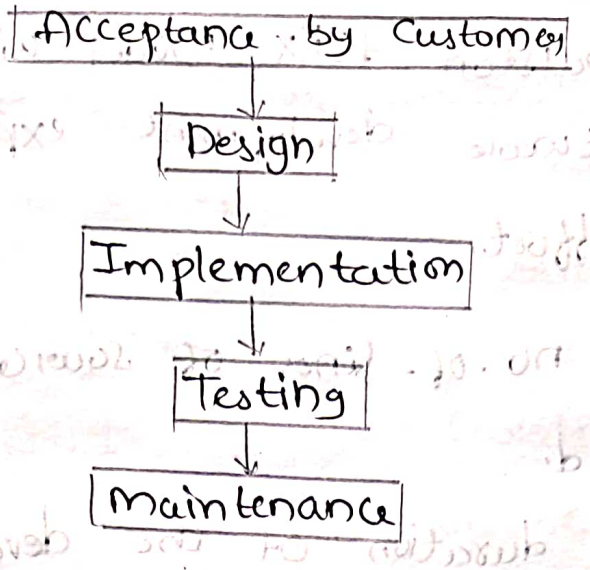
3rd Incremental level:-

Advanced Page layout.

Prototype Model :-

* A prototype is a toy implementation of the system.
* Prototype model requires that, before carrying out the development of Actual software.
* A working Prototype of the system should be built.





Where to use prototype model:-

It is used when lot of interaction with the end user. It is used for online systems and web interface

Compare the walston felix model with the SEL Software engineering laboratory model on a software development expected to involve 8 person years of effort.

a) Calculate the no. of. lines of source code that can be produced.

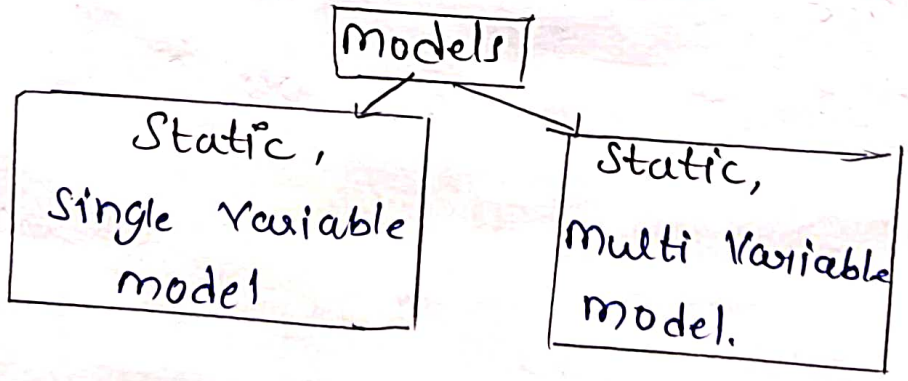
b) Calculate the duration of the development

c) Calculate the productivity in LOC/py and Calculate the average manning.

Cost Estimation model:-

Model may be static or dynamic.

- In static single variable is taken as key element of calculating the cost and time
- In dynamic model all variables are interdependent and there is no basic variable.



* The common static of the both static model is $C = aL^b$

C = Cost

L = Size

a & b are constants.

Static: Single Variable model using SEI model:

$E = 1.4 L^{0.93}$

$DOC = 30.4 L^{0.90}$

$D = 4.6 L^{0.26}$

E = Efforts [percent per month]

DOC = Documentation [No. of Pages]

D = Duration [month]

L = No. of lines per Code

Static: Multi Variable model Using SEI Model:

It gives the relationship between lines of source Code and effort.

$E = 5.2 L^{0.91}$

$D = 4.1 L^{0.36}$

$I = \sum_{i=1}^{29} w_i x_i$

w = weight

i = variable

Xi = Variable decreasing

Xi (-1, 0, 1)

a) 8.1 years of effort = 96 months

$E = 1.4 L^{0.93}$

$L(SEI) = \left(\frac{96}{1.4}\right)^{1/0.93}$

$= [68571.4]^{1.075268}$

$= 94.264$

$= 94264$ LOC

$E = 5.2 L^{0.91}$

$L = \left(\frac{E}{5.2}\right)^{1/0.91}$

$= \left(\frac{96}{5.2}\right)^{1/0.91}$

$= (18.4615)^{1.098}$

$= 24.567$

$= 24632$ LOC.

b) Duration in months can be calculated by means of equation.

(i) $D = 4.6 \, L^{0.26}$

$= 4.6 (94.264)^{0.26}$

$= 4.6 (3.26)$

$= 14.996$

$D = 15$ months.

(ii) $D = 4.1 \, L^{0.36}$

$= 4.1 (24.632)^{0.36}$

$= 4.1 (3.16)$

$= 12.9$

$D = 13$ months

c) productivity in Loc/py.

(i) Single variable

$$I = \frac{LOC}{PY}$$

$$I = \frac{94.264}{8}$$

$= 11.783$

$= 11783$ LOC

(ii) Multi Variable.

$$I = \frac{\text{Lines of Code}}{\text{person year}}$$

$$I = \frac{24.632}{8}$$

$I = 3.079$

$I = 3079$ Loc

d) Average Manning:-

Average Manning $= \frac{E}{D}$ persons.

Single Variable

$E = 96$, $D = 15$ months

$= \frac{96}{15} = 6.4$

$= 6.4$ persons

Multi variable

$E = 96$, $D = 13$ months.

$= \frac{96}{13} = 7.38$

$= 7.4$ persons.

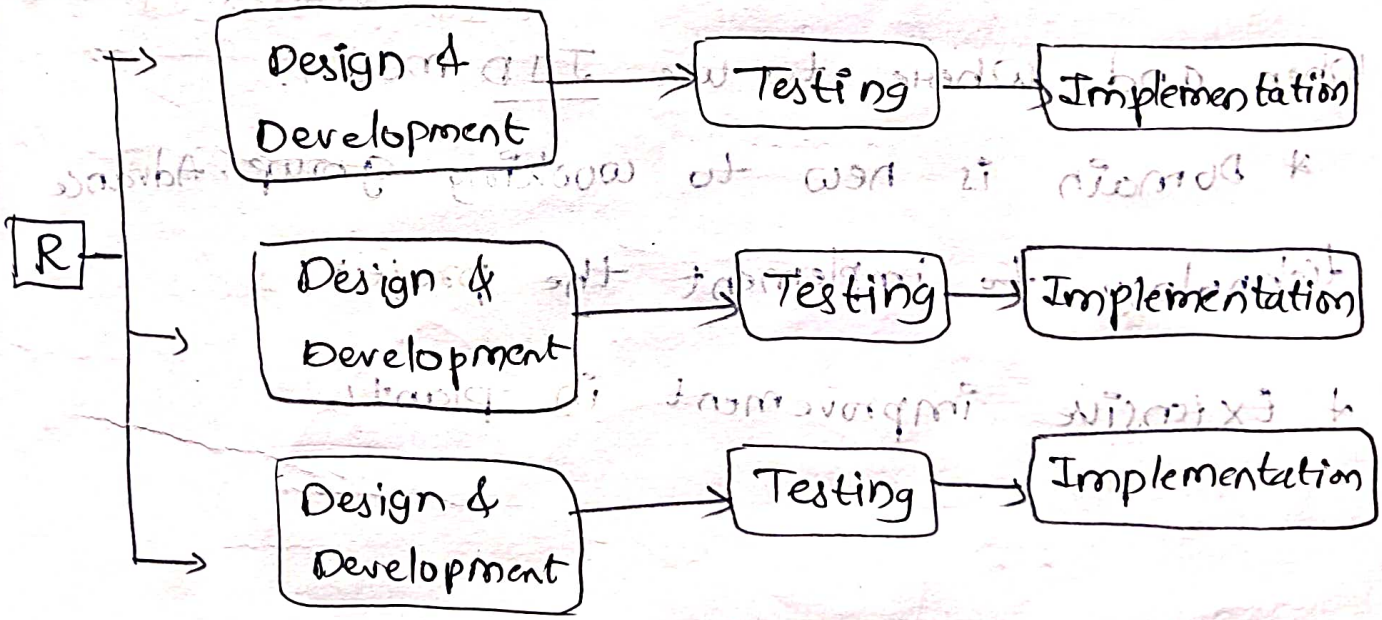
~~ITERATION~~

PROCESS ITERATION & ACTIVITIES:-

IID is a model developed for multiple cycles for iteration. During this process Developer has a advantage of evaluating, and Testing at each stage.

* Iterations include update and execution of cycle are to be basic, direct and particular

* It starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving sessions.



4 phases:-

- Inception
- Elaboration
- Construction
- Transition.

Inception:-

In this phase project scope and risk of each stage is Analysed.

Elaboration:-

In this phase Conveying / Converting working model into a map.

Construction:-

In this phase Coding and Architecture built.

Transition:-

In this phase project is converted into production and then deployed.

When and where to use IID (Iterative & Incremental development) model?

* Domain is new to working group, Advance technology to implement the project.

* Extensive improvement in plants

RATIONAL UNIFIED PROCESS:-

- It is a Software development Process model for object oriented model.
- It is also called as UML- Unified Modelling Language.
- It is created by rational. Operation and design and documented using UML.
- RUP is a Software Engineering process, delivered through a web-enabled searchable knowledge base.

Production:-

* project is maintained and updated.

Advantages:-

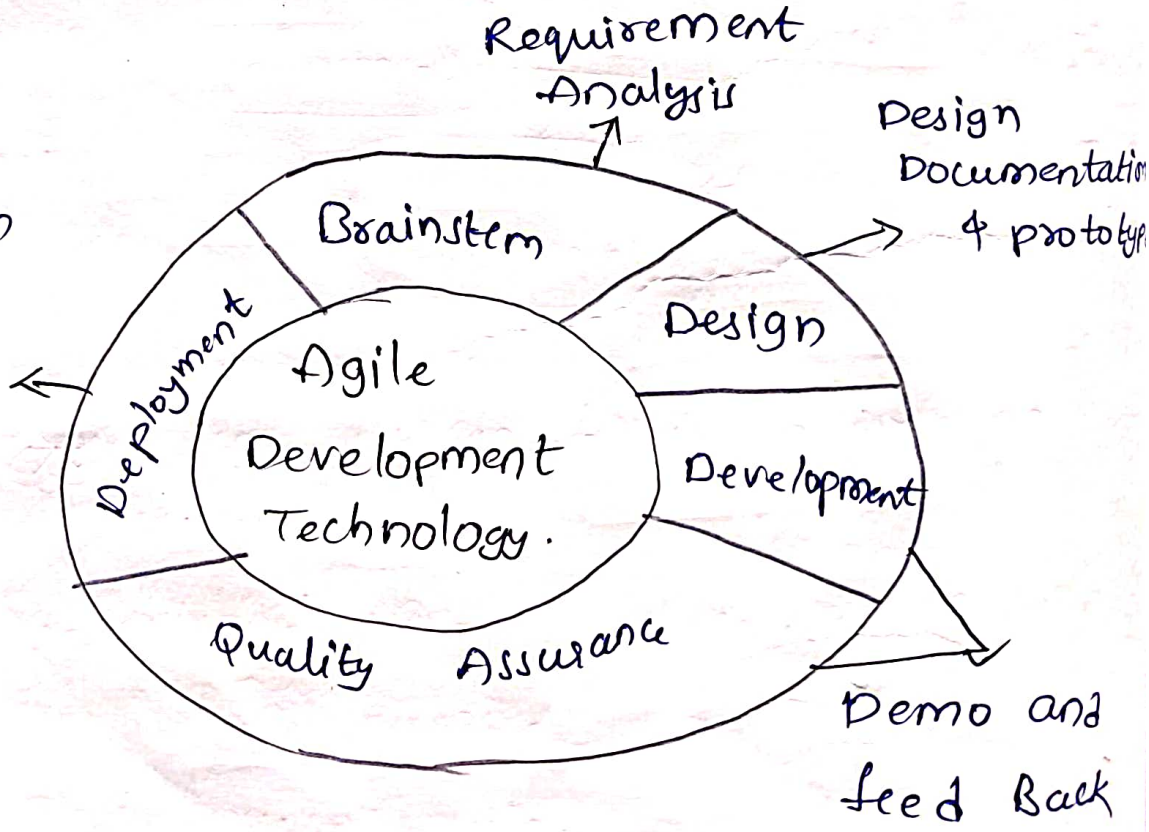
- No risk management support.
- Reusable Components.
- Less time Duration
- It provides good documentation.

Disadvantages:-

- process is Complex
- Hard to integrate again and again

Agile View of process:-

- Agile process is the software development approach based on iterative development.
- Agile method breaks task into smaller iteration.

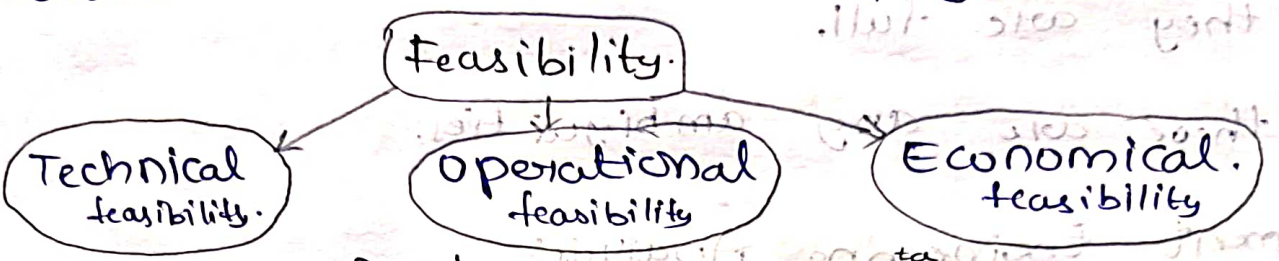


Requirement Engineering Analysis:-

Requirement engineering refers to the process of defining, documenting and maintaining requirements in the design process.

Feasibility Study:-

- create a reason for developing a software.



Requirement Analysis and Elicitation:-

- To Collect the information.

Problems of Elicitation and Analysis:-

- Getting all and only the right people involved.
- Stack holders often they do not know what they want.
- Stack holders ~~respond~~ express requirements in their terms.
- Stack holders may have conflicting requirements.

User and System Requirement:-

- Document is created by Analyst.
- In this model include 3 stages.

ER Diagram (Entity - Relationship Diagram)

Data flow Diagram

Function Decomposition diagram.

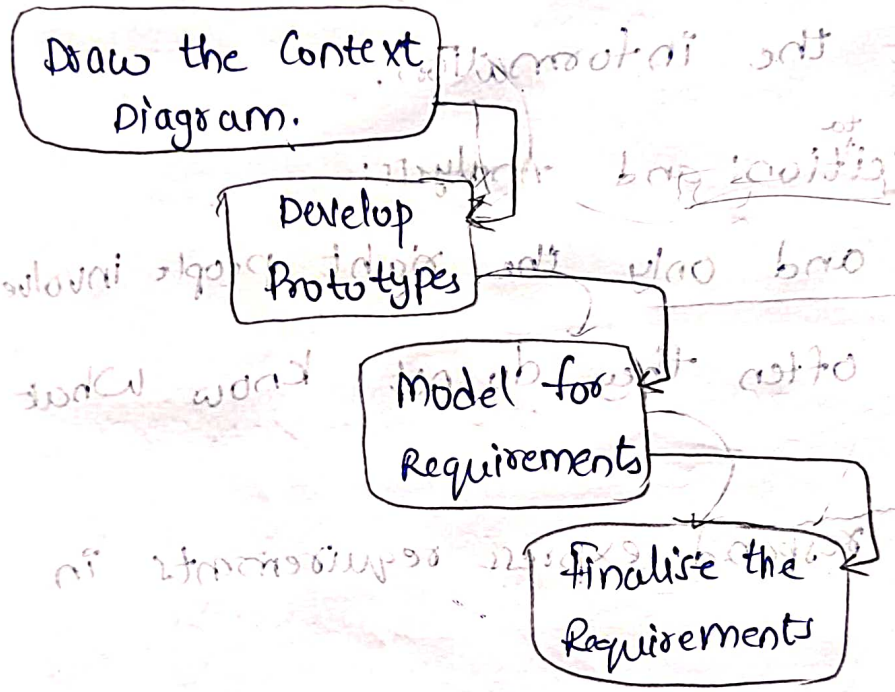
Requirement Validation:-

- The documents are validated in this pha

*Criteria

- They can be practically implemented.
- If they can describe.
- If they are full.
- If there are any ambiguities.

Requirement Engineering Analysis:-



Software Requirement Specification:-

Here are five steps you can follow to write an effective SRS document.

1. Define the purpose with an Outline (use an SRS Temp
2. Define your Products purpose.
3. Describe what you will build.
4. Detail your specific Requirements
5. Deliver for Approval.

Advantages

* simplicity and lack of bias.
* minimizes the time and effort required by developers to acheive desired goals.

Disadvantages.

* The time needed to gather the full list of a specific population.
* The capital necessary to retrieve and contact that list.

22/3/23

COCOMO MODEL:- (Constructive Cocomo Model).

* It is a Constructive Cost model.
* It was designed and proposed by Barry Bohm in 1991.
* It is divided into Basic model, Intermediate model and detailed model.

Basic Model:-

* It is the first level that can be used for Quick and slightly rough calculations of Software Cost.
* It requires to calculate the efforts.
* It can be broadly divided into three types
  - Organic
  - semidetached
  - Embedded.

BASIC COCOMO MODEL:-

$E = a^* (KLOC)^b$

$D = c^* (effort)^d$

$P = \frac{Effort}{Time}$

E = Effort (applied in per month)

D = Development (Time in month)

P = No. of persons Required for the project.

Constant values a, b, c, d for the basic model for the different categories of the system.

Software Project	A	B	C	D
organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Effort:-

$A = a^* (KLOC)^b$

$= 3.0 \ (300)^{1.12}$

$= 3.0 \times 594.806$

$E = 1784.418$

Development:-

$D = c^* (effort)^d$

$D = 2.5 \times (1784.418)^{0.35}$

$D = 2.5 \times 13.74$

$D = 34.35.$

(iii) $P = \frac{\text{effort}}{\text{time}}$

$P = \frac{1784.418}{34.35}$

organic

$A = a * (KLOC)^b$

$= 2.4 (300)^{1.12}$

$= 2.4 (399.0)$

$= 957.6.$

$D = C * (\text{effort})^d$

$D = 2.5 (957.6)^{0.38}$

$D = 2.5 \times 13.578$

$D = 33.945$

$P = \frac{\text{effort}}{\text{time}}$

$P = \frac{957.6}{33.945}$

$P = 28.21$

Embedded:-

$A = a * (KLOC)^b$

$= 3.6 (300)^{1.20}$

$= 3.6 (938.7)$

$= 3379.3$

$D = C * (\text{effort})^d$

$D = 2.5 (3379.3)^{0.32}$

$D = 13.4 \times 2.5$

$D = 33.5$

$P = \frac{\text{effort}}{\text{time}}$

$P = \frac{3379.3}{33.5}$

$P = 100.87$

There are 6 phases in Cocomo model.

1. Planning and Requirement
2. System Design
3. Detailed Design.
4. Module Code and Test
5. Integration and Test
6. Cost Constructive model.

Advantages

- Can be used to estimate the cost and time and effort for the software project.
- Helps in identifying cost, effort, time. with high impact.
- Can be used to evaluate the feasibility.

Disadvantages:-

* size of the software which may not always available in this case.
* Doesn't provide an accurate estimation of cost, time and effort.

Cocomo MODEL 1

* It is useful in waterfall model of SDLC.
* It provides estimation for time, effort and cost.
* It is based on linear Reuse formula
* It has three development modes.

Cocomo MODEL 2

* It is useful in non sequential, Rapid development and Reusable model of s/w
* It provides estimation for standard deviation
* It is based on Non-linear Reuse formula.

* It begins with Requirement s/w

* size of the software depends on LOC.

* It has 3 development modes.

* It begins with / follows Spiral type of development.

* size of the software depends on function points Object points & LOC.

Intermediate Cocomo Model:-

This model recognise the initial estimation for set of 15 cost drivers based on various attributes in software engineering.

Classification of cost driver & their Attributes.

* Product Attribute
* Hardware Attribute
* Personal Attribute
* Project Attribute.

Formula:-

$E = a_i (KLOC)^{b_i} * EAF$

$D = c_i (E)^{d_i}$

EAF = Effort Adjustment factor.

Software Project	a_i	b_i	c_i	d_i
Organic	2.4	1.05	2.5	0.38
semi-detached	3.0	1.12	2.5	0.35
embedded	3.6	1.20	2.5	0.32.

for a given project was estimated with a size of 300 kloc. Calculate the effort schedule time for development. By considering the developer having high application experience and very low programming experience.

$$E = a_i (KLOC)^{b_i} EAF P_m$$

$$\frac{E}{a_i (KLOC)^{b_i}} = EAF P_m$$

Organic System:-

$$\text{Effort} = 2.8 \times (300)^{1.05}$$

$$= 2.8 \times (399.0)$$

$$= 1117.21$$

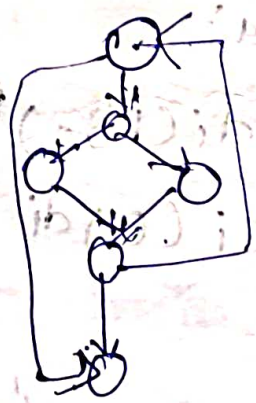
$FP = 500$

$\frac{LOC}{FP} = 50$

$LOC = 500 \times 50$

$LOC = 25000$

$KLOC = 25$



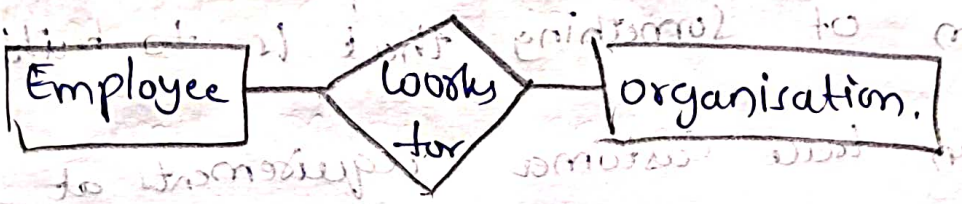
$E = a_i (KLOC)^{b_i}$ EAFPm

$\frac{E}{a_i (KLOC)^{b_i}} = EAFPm$

$= \frac{E}{2.4(300)^{1.05}} = \frac{117.21}{2.4 \times 399.0} = \frac{1117.21}{957.6}$

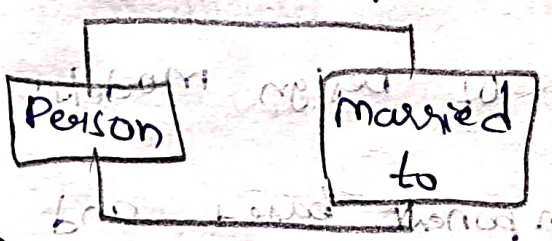
$= 1.16667.$

RELATIONSHIP:-



Types:-

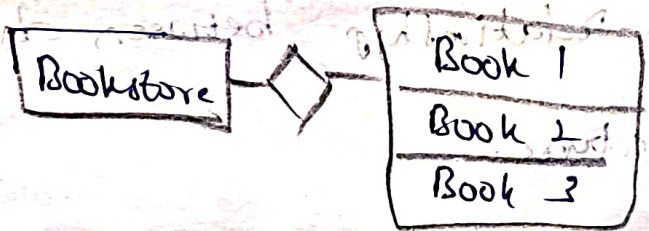
Unary Relationship



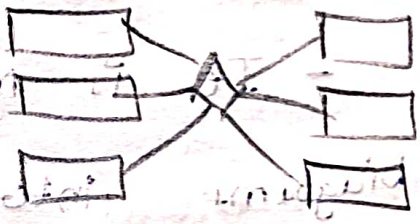
Binary Relationship

(i) one to one

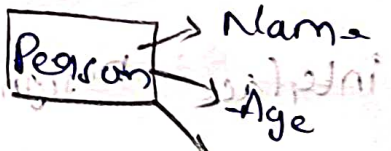
(ii) One to many



(iii) Many to many



Ternary Relationship



SOFTWARE DESIGN:-

-specifying, designing, implementing, validating, Deploying and maintaining.

- The system engineering process comes under waterfall model.

What is Design?

-Design is a meaningful engineering representation of something that is to built.

- It can trace customer requirements at the same time, Quality, Good Design.

- It is mainly used in data flow, Architecture Diagrams, interfaces and components.

* There are four types of Design models:-

Data, Architecture, Component based and interface Design.

Data Design:-

* Creates a model to implement the software.

Architecture Design:-

- It defines the relationship between design Pattern and architecture.

Interface Design:-

It describes how the software communicates within itself.

Component Based Design:-

* It is a procedural Description.

* Software Design is an iterative process through which the requirements are transferred into blueprint for developing a software.

* The main aim of design is to generate a model which shows fitness, Delight and Commodity.

* Characteristics for good design:-

- A design should be modular.

- Design should be exhibit in a architectural structure using evolutionary fashion. Components and recognisable design pattern.

- A design should contain data, architecture interface and components.

- A design should be readable and should understandable.

Design Quality:-

- Design" a root for all qualities include Quality of the product service experience

Quality of the system and process.

* Designer can improve the reliability by ensuring the software is easy to implement & change.

1. TOP DOWN APPROACH:-

* Top down design takes the whole software system as one entity to acheive more than one sub system or components. Based on characteristics.

* This process keeps on running until the lowest level of system in the top down hierarchy is acheived.

* Top down design is more suitable when software solution need to be designed from the scratch and specific details are unknown.

2. BOTTOM-UP APPROACH:-

* Bottom up design model start with most specific and basic components.

* It keeps creating high level components until desired system is not evolved with the each higher level. The amount of abstraction is

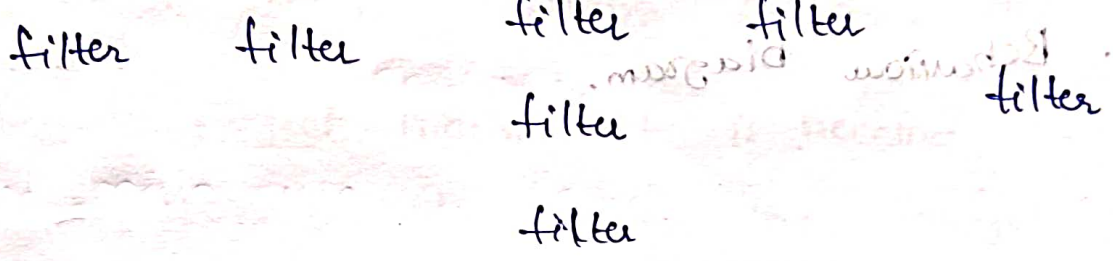
Bottom up strategy is more suitable for Creating the existing model.

Architecture Design:-

DATA FLOW ARCHITECTURE:

1. It is of two types.

(i) PIPE AND FILTER PATTERN:-



(ii) BATCH SEQUENCE PATTERN:-

(i) Set of Components called filters which are connected by Pipes to transfer data from one Component to another Component.

(ii) This pattern accepts the batch of data and this applies a series of sequential components to transform it.

CALL AND RETURN ARCHITECTURE:-

It enables to achieve a program software / structure which is relatively easy to modify and scale.

Waterfall model
Topdown
Cmm.

UML:- (UNIFIED MODELLING LANGUAGE).

* It is a rich language to model software solutions application structure. System behaviour and Business process.

There are two main categories in UML Diagram.

1. Structured Diagram
2. Behaviour Diagram.

① STRUCTURED

- Class Diagram
- Component Diagram
- Deployment Diagram
- Object Diagram
- Package Diagram
- Profile Diagram
- Composite Structure Diagram.

② BEHAVIOUR

- Use CASE Diagram
- Activity Diagram
- State - machine Diagram
- Sequence Diagram
- Communication Diagram
- Interaction Overview Diagram
- Timing Diagram.

23/3/23:-

SPM (Software project management)

Software project management is a proper way of planning and leading the software project.

* Software is a non physical product.
* Software is developed for business and with the little experience in building software product.
* Software project management is necessary to incorporate users requirements along with the budget and time constrain.

Quality

Quality defines measurable characteristics like Correctness, Maintainability, portability, Testability, Usability, Reliability, Efficiency, Reusability and Interoperability.

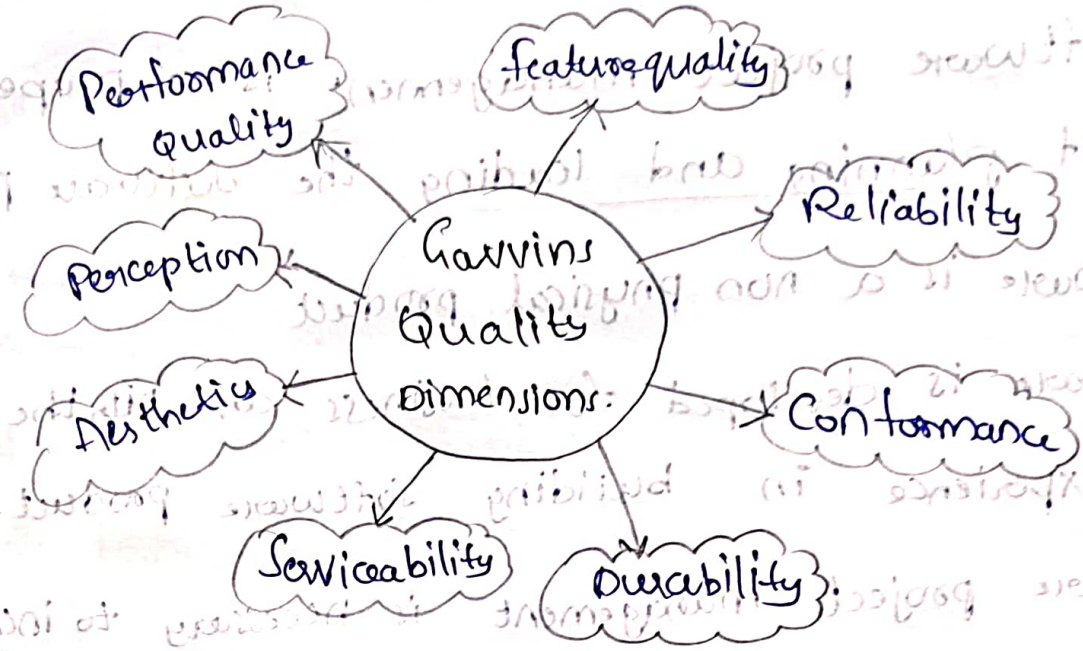
There are two types of quality:-

- Quality of design
- Quality of Conformance

Software Quality:-

It is defined as the conformance to explicit state function and performance evaluation.

Garvins Quality dimensions:-



Performance Quality:-

It check whether the software system is delivered with all the contents, functions and Option that are addressed during the project demo

Feature Quality:-

Software system offers Option that surprise and delight for first time user.

Reliability:- It will check whether the software system is delivered with all the contents, functions and option that are addressed during the project demo. it will not failure.

It will deliver error free product.

Conformance:-
- will the software system adjust into native/internal and external software standard.
* Factual style and writing conventions.
* will the computer program accepts the rules.

Durability:-
- will the software system can be maintained & changed?
- can be reduce the error rate within the time period.

Serviceability:-
- will the employee support if any corrections errors or changes.

Aesthetics:-
Novel flow and clear presentation.
(uniqueness)

Perception:-
If you are introduced to a software project that has been created with poor quality, in past, presents of software product quality may be influenced negatively.

Mccall's Quality factors:-

- This model was introduced in 1977 by Jim Mccall's.
- The Quality model is divided into 11 factors.
- perspectives of Mccall's Quality is divided into 3 types as a model

1. Product Operation factors
2. Product Revision factors.
3. Product Transition factors.

①. PRODUCT OPERATION FACTORS:-

- It checks whether software able to perform or not

- **Correctness:-**

Effort with which software perform according to the user requirement.

- **Usability:-**

- The effort with which software program are understood and operated.

- **Efficiency:-**

Capability of the software to perform with Optimal Resource

- **Reliability:-**

Effort with which the software is expected to perform Speitic task accurately.

. Integrity:-
unauthorized access to the software. Can be Controlled.

② PRODUCT REVISION PHASE:-
- It checks whether software needs any modification.
(How the system can be Tested, changed or re-deployed.

• Maintainability:-
The effort with ~~Eaith~~ which software can be easily modified.

• Testability:-
To Test a software, ensure that it executes the accurant result

• Flexibility:-
The effort with system software or program can be modified.

③ PRODUCT TRANSITION PHASE:-
It checks whether the software is adaptable to the new environment or not.

Portability:-
- The effort to transfer a software from one hardware/software to another hardware/software environment.

Reusability:-
A program can be reused in other applications.

Interoperability:-

The effort with which a system integrates with another system.

Software Testing:-

Testing:-

Testing is a process of executing program with the intent of finding the error.

White-Box Testing:-

White box Testing is a Test Case design that uses the Control Structure of procedural Design of a Software. White-box Testing is also called as ~~Functional~~ Structural Testing.

Black-Box Testing:-

Black-Box Testing is a Test Case design method that focused on functional Requirements of the Software. Black-Box Testing also called as Functional Testing.

WHITE - BOX TESTING

- No knowledge of the internal logic of the system is used to develop Test cases.
- Tents to uncovers errors. That occurs in implementation (or) design Requirements.
- Uses Validation techniques
- It applied during the later stages of testing

Ex:- Unit Testing, Integration Testing, System Testing and Acceptance Testing.

BLACK - BOX TESTING

- Knowledge of the internal logic of the System is used to Develop Test cases.
- It Tents to Cover errors - That occurs in implementation (or) Design Requirements.
- And it Uses Verification Techniques.
- It is applied during the earlier stages of Testing.

Ex:- Feasibility, Reviews, Requirement Reviews.

Verification:-

Verification refers to set of Activity that ensure whether the software currently implementing a specific (that) function.

Validation:-

Refers to the set of Activity that ensure that the software has been built is traceable to the customer Requirement.

Types of Testing:- (Functional Testing).

1. UNIT TESTING:-

It helps to identify the bugs earlier in the development process.

*It is expensive to use:-

Ex:- It checks whether loop method or function is working on time. I checks the incorrect initialization.

2. Integration:- (Refer Concept Map)

3. Smoke Testing:-

This type of Testing is done to check whether the Testing is ready or stable for further Testing.

Ex:- If the product has two modules before going into that module make sure the module 1 is working properly.

4. Interface Testing:-

Interface Testing is Type of Testing which verifies whether communication between two different software is done correctly.

• Connection that integrates two components are called Interface

Ex:- Suppose for any XYZ application, the interface states ATM S as an input and delivers

JSON is as a output.

5. Usability Testing:-

It is a Testing method for measuring how easy and user friendly the software is.

6. System Testing:-

System Testing is actually a series of different Testing whose primary purpose is to fully exercise the Computer base system.

It verifies whether the system element have being properly integrated and performed the allocation functions.

1. Compatibility Testing:- (Non functional Testing)

It is the type of testing to ensure that a software program (or) system is capable with other software program or system.

2. Load Test:-

It is the type of Testing to ensure that a software program (or) system need a specific compliance standard.

Ex:- HIPAA.

3. Performance Testing

It is the type of Testing to ensure that a software program (or) System needs the

Specific performance (or) goal.

Ex:- Response time,

Security Test:- It is the type of Testing to ensure that a software program (or) system needs secures from unauthorized access or attack.

Scalability Test:- It is the type of Testing to ensure that a software program/system can be scaled up or down to meet the changing needs.

* Likewise we have recovery testing, Volume testing, Stress testing, Usability testing, Compliance testing.

WHITE BOX TESTING:-

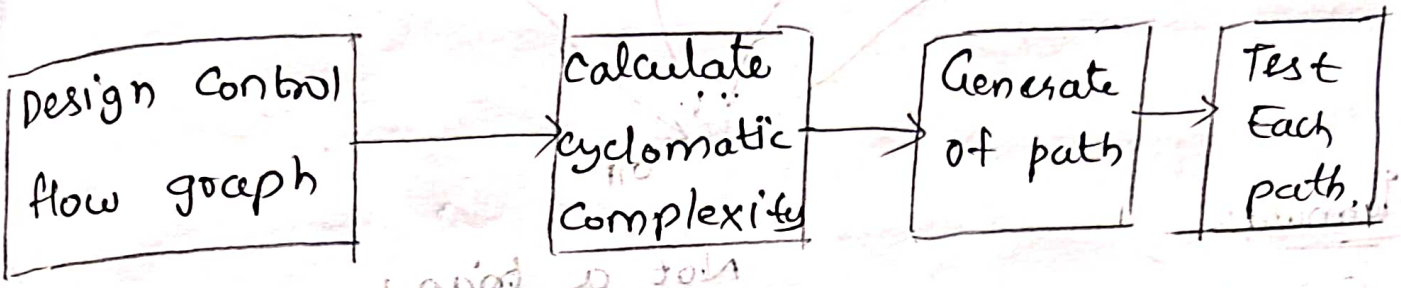
BASIC PATH TESTING:-

Basic path Testing is one of the Structural Software Testing methodology. It is also called as Source code Based Testing. It uses control flow graph Based on ~~Chromatic~~ Cyclomatic Complexity.

Cyclomatic Complexity = $E - N + 2$

E - Edges, N - Nodes

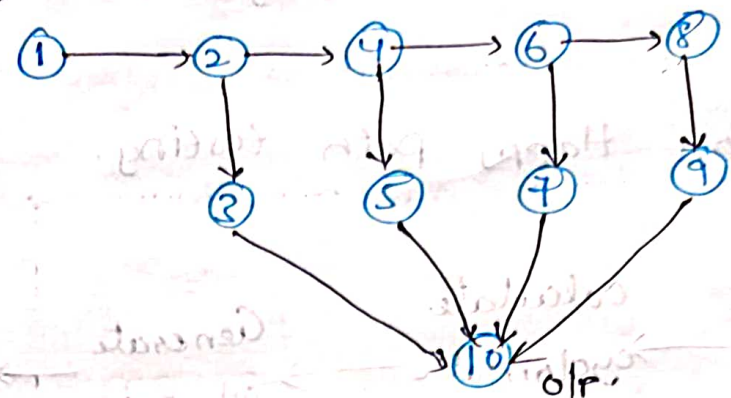
* It generates test.
* It is based on path testing vs branch testing.
* It is Based on Happy path testing.



Triangle problem:-

1. Input a,b,c
2. if (a>=(b+c) or b>=(a+c) or c>=(a+b)) then
3. output = "Not a triangle"
4. Else if (a==b and a==c) then
5. Output = "Equilateral Triangle"
6. Elseif (a==b or a==c or b==c) then
7. Output = "isosceles triangle"
8. Else
9. Output = "Scalene Triangle"
10. Return Output.

Control flow graph:-



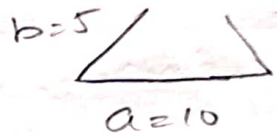
Formula:-

$CC = E - N + 2$

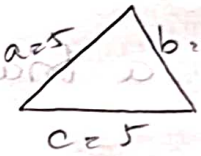
$CC = 12 - 10 + 2$

$CC = 4$

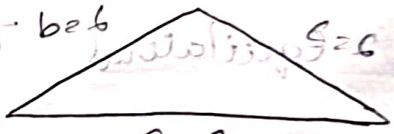
Not a triangle



Equilateral Triangle.



Isosceles Triangle



BLACK BOX - TESTING.

EQUIVALENT PARTITIONING:-

- Test design Techniques.
- Divide Test Data Techniques.
- Divide input Test Data into partitions
- Test each partition once. ((The assumption is that the any input within a partition is ~~equilent~~ equivalent.)

produces same output)).

Example 1:- Date (1 to 31)

Invalid Testing	Valid Testing	invalid Testing
-1	1 6	32
0	2 7	33
	3 8	34
	4 9	
	5 ... 31	

. User Name (6 to 8) character.

invalid Testing	Valid Testing	invalid Testing
character 0 to 4	character 5 to 8	Character 9 to ∞.

Age (18 to 80 yrs) Except 60 to 65 yrs.

invalid Testing	Valid Testing	Valid Testing	invalid Testing
0, 1, 2, 3	18, 19, 20	60, 61, 62	66, 67,
.....	.....	63, 64, 65	.....
.....17	59		80.

Boundary Value Analysis.

* It is a test design technique.
* This technique is ~~equivalent~~ related to equal partitioning
* Boundary Value Analysis Test Both sides of each Boundary.
* The assumption is that the system behaves differently on either sides of the boundary

Invalid partitioning - Valid partitioning		Invalid partitioning - Valid partitioning	
Lower - value Boundaries		Upper value - Boundaries.	
Invalid	Valid	Invalid	Valid
0	1	32	31
Username (5 to 8) characters			
Invalid	Valid	Invalid	Valid
0	5	9	8
Floating point Values (0.2 to 0.8)			
Invalid	Valid	Invalid	Valid.
0.1	0.2	0.9	0.8

MANAGEMENT:-

Risk management is an important part project planning Activity.

Risk management involves identifying and limating the probability of risk with their ct of impact on project.

eeds for Risk management System:-

* To determine the Risk and the factor at causes them.

All the Risk should be classified and ioritized.

Create a plan that connects and creates each isk to a migration strategy.

Throughout a project keep an eye on Risk Trigger. If risk occurs Take the appropriate migration measures. Through out the project Communicate the status of Risk.

* Risk Management can be broadly divided into 4 categories.

1. Identification:-
2. Risk Analysis
3. Risk Planning
4. Risk monitoring

1. Risk Identification:-

* Risk Identification involves Brain storming activity.
* Brain storming is a group of discussion technique where all the stack holders meet together.
* Risk identification technique produces new ideas a promotes creative thinking.

2. Risk Analysis and prioritize:-

* Identifying the problem causing risk in the project
* Identify the probability of occurance of problem
* Identifying the impact of problem.

RISKS	Category	Probability	impact
Size estimate low	PS	60%	2
More users than planned	PS	30%	3
Less reuse than planned	PS	70%	2
End-users resist system	BU	40%	3
Delivery Deadline tightened	BU	50%	2
Funding will be lost	CU	40%	1
Customer will change Requirement	PS	80%	2
Technology not meet expectation	TE	30%	1
Lack of training on tools	DE	80%	3
Staff inexperienced	ST	30%	2

Risk Avoidance and migration: (planning).

. The purpose of the Technique is to avoid or iminate the occuranc of risk.

. This method used to avoid risk. To reduce e scope of project by removing non essential equirements.

. Risk Monitoring:-

. In this technique the risk is monitored ontinuously by Revaluating (reconstruction) the risk. the impact of risk and probability of occurance of Risk.

22/5/23

SCRUM:-

* Scrum is a type of Agile framework.

* It is a framework within which people can address complex adaptive problem while productivit and creativity of delivering product at highest possible values.

Features of SCRUM:

* SCRUM is a light weight framework.

* SCRUM can manage and control software and product development.

* SCRUM uses iterative and incremental model.